Pilotage de VLC à l'aide d'un téléphone Bluetooth

Jérémie Albert, Julien Maillard, Jérôme Verlhac, Nicolas Thomas

January 12, 2007

1 Objectifs

L'objectif de ce projet est de piloter le logiciel VLC (VideoLAN Client) à l'aide d'un téléphone Bluetooth qui permet notamment une lecture de fichiers multimédias. Nous souhaitons au final être capable d'effectuer des opérations simples telles que la lecture d'un fichier multimédia, la lecture d'une playlist, la mise en pause, le changement de piste et le contrôle du volume.

2 Reprise de l'existant

Côté téléphone, la communication Bluetooth a été reprise sur le projet 7 de l'an dernier. En effet, celle-ci fonctionnant bien, il n'y avait pas de raison de la réimplementer.

Côté serveur, la partie communication est fortement inspirée du tutorial C nommé "An Introduction to Bluetooth programming in GNU/Linux".

3 Travail effectué

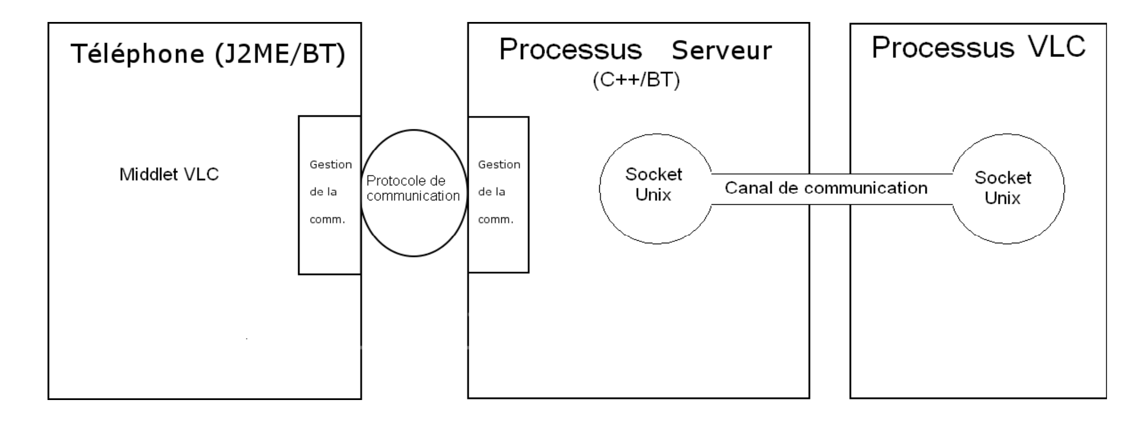


Figure 1: Architecture simplifiée du projet

3.1 Téléphone

Nous avons développé un midlet J2ME afin de piloter VLC depuis un téléphone Bluetooth.
Les éléments de l'interface du téléphone sont contenus dans un Form. Ce Form est créé

en fonction de booléens qui nous permettent de déterminer l'état de l'application, c'est à dire si un fichier est chargé ou non et si une erreur de connexion lors de la communication avec le serveur a eu lieu. Selon les cas l'affichage sera différent.
Les différents éléments contenus dans notre interface sont des objets StringItem, TextField et Command. Les StringItem peuvent être de simples chaînes de caractères ou des chaînes contenues dans un bouton auquel on pourra associer une action. Les TextField nous permettent de créer des champs pour la saisie de caractères. Les Command permettent d'ajouter des boutons au menu déroulant de la midlet comme le bouton "exit". Pour ajouter une fonctionnalité supplémentaire, il suffit de créer un nouvel item et de lui associer l'action appropriée selon le même schéma que les autres items. Par contre, si la fonctionnalité concerne le serveur, il ne faut pas oublier de mettre en place l'envoi du message approprié depuis le téléphone et de le traiter sur le serveur.

Par contre, au niveau du menu déroulant du midlet (lorsque l'on clique sur VLC lors de l'exécution), nous avons mis en place une fonction exit permettant de sortir de l'application. Cependant, un problème perdure: à chaque fois que l'on recréé un Form, un Exit supplémentaire est ajouté au menu. Pourtant, avant la création des différents éléments de l'interface, on supprime tous les anciens éléments en exécutant la ligne "choix.deleteAll();" dans la méthode createForm() de VLC.java. Le problème vient probablement du fait que cette méthode ne supprime pas les objets "Command" présents dans les objets Form. Par manque de temps, nous n'avons pas réglé ce problème. Mais cela n'empêche pas le bon fonctionnement de l'application.

3.2 Serveur

Le serveur est implémenté en C++.

L'interface de communication Bluetooth avec le téléphone se fait via le module BTConnection. Il initialise un socket serveur permettant d'attendre la connexion d'un client. Actuellement, ce module permet de gérer seulement un client à la fois. Ainsi, lorsque qu'un client s'est connecté, les méthodes receiveMsg et sendMsg servent à communiquer directement avec ce client. Par ailleurs, lors d'une écoute dans receiveMsg, si le client se déconnecte, cela est detecté. On retourne alors un message nul afin que le serveur puisse se rendre compte de la situation et réagir en conséquence.

Module de communication BTConnection.

Au démarrage, on lance un processus pour exécuter VLC. La communication entre le serveur et VLC est effectuée grâce à une socket Unix. En effet, VLC possède un mode serveur qui lui permet d'attendre des commandes depuis une socket Unix plutôt que depuis une interface graphique. Pour cela, il est nécessaire de le lancer avec les arguments suivants: "-I rc –rc-unix filename" où filename est le nom de fichier faisant référence au socket Unix. De cette manière, l'envoi d'une commande telle que "pause\n" met en pause la lecture de la piste en cours. L'ensemble des commandes lui est donc passé de cette manière. Il faut noter qu'un backslash est nécessaire pour que le buffer du socket soit immédiatement vidé. Ainsi la commande est aussitôt prise en compte par VLC. De plus, nous renvoyons directement le message du téléphone à VLC. Le serveur ne fait que du "forwarding". Voici quelques commandes possibles:

- add XYZ: charge le fichier XYZ et lance la lecture. XYZ peut être un chemin relatif ou absolu.
- stop: arrête la lecture du fichier en cours et revient à l'écran de chargement.
- pause: basculer la pause
- f: basculer le mode plein-écran
- volup X: augmenter le volume X.
- voldown X: diminuer le volume X.

Ces commandes correspondent aux message que le téléphone envoi.

4 Manuel d'utilisation

4.1 Matériel et librairies nécessaires

- Librairie libBluetooth (testé avec la version 2-1.7-1).
- Téléphone Bluetooth.
- PC sous linux avec connexion Bluetooth.

4.2 Installation

4.2.1 Téléphone

Le midlet du téléphone a été compilé avec le J2ME Wireless Toolkit de Sun. Il faut donc installer le fichier jar sur le téléphone.

4.2.2 Serveur

Le programme VideoLan Client doit bien sûr être installé sur la machine. De plus l'exécutable nommé "vlc" doit se trouver dans un des chemins de la variable d'exécution PATH. Pour compiler le serveur, il suffit de lancer la commande make. L'exécutable obtenu est "main".

4.3 Lancement du programme

Tout d'abord, il faut lancer le serveur:

```
>./main
```

Une fois que le message "Interface de commande à distance initialisée, h pour obtenir de l'aide" apparait dans le terminal, le serveur est prêt à recevoir une connection. On peut alors lancer l'application du téléphone.

4.4 Actions possibles

Au demarrage de l'application du téléphone, un menu apparaît et nous propose de charger par defaut le fichier 1.mp3 present dans le repertoir de main. Il suffit d'éditer ce nom afin de charger le fichier souhaité. Une fois chargé, un nouveau menu aparait. On peut alors mettre en pause (bouton pause), passer en mode plein écran ou en sortir (bouton screen), augmenter (bouton vol up) ou diminuer (vol down) le volume ou encore sortir de mode lecture et revenir à l'écran de chargement en cliquant sur le bouton stop. our information vol up et down incrementent ou décrementent le niveau du volume selon une valeur qui est crée en dur.

5 Conclusion

5.1 Avantages/Inconveniants

+ Maintenabilité: grâce à des modules indépendants comme BTConnection par exemple.
- Dépendance par rapport à VLC.

5.2 Evolutions

Une évolution intéressante d'un point du vue interactif avec le téléphone serait de récupérer les informations du serveur sur le téléphone. La plus importante serait la récupération des erreurs de VLC (comme l'impossibilité de charger un fichier). On pourrait aussi recupérer l'état d'avancement de la lecture ou le niveau du volume. De plus, pouvoir naviguer dans le système de fichier depuis le téléphone faciliterait le chargement d'un fichier.

Par ailleurs, nous n'avons pas mis en place de services utilisant SDP (Service Discovery Protocol) coté serveur ni la découverte de tels services coté client. Ceci permetrait d'éviter de coder en dur l'adresse mac du serveur.

Une autre évolution possible serait de pouvoir gérer plusieurs clients simultanément. Pour cela, dans la classe BTConnection, il suffirait de gérer les clients sous forme de listes (plus particulièrement sous la forme d'une liste de sockets). A partir de là, il faudra modifier les méthodes d'envoi (sendMsg) et de reception (recvMsg) de messages afin de choisir avec qui on souhaite correspondre. Cette évolution pourra permettre de piloter plusieurs VLC simultanément à partir d'un seul serveur. Cela peut être intéressant car il est possible de lancer VLC sur plusieurs machines et de les piloter via le réseau par connexion TCP.

Enfin, une dernière amélioration consisterait à améliorer l'interface en organisant les éléments de faon à faciliter l'utilisation de l'application côté téléphone.